2.5 RADAR SENSOR

The radar sensor class is modeled using the radar range equation with jamming when the model is used to detect aircraft or TM targets. The radar sensor class uses a detection probability to determine the detection of ground targets.

2.5.1 Functional Element Design Requirements

Not currently available.

2.5.2 Functional Element Design Approach

The equations and algorithms which are used to model the behavior of the radar can be found in the EADSIM Methodology Manual, Version 5.00 [2]. The specific sections to reference, in the order discussed in the next section (2.5.3 Calling Tree) are:

TABLE 2.5-1.

Section	Calling Tree Procedure(s)
6.5.1	RangeRateGate, RangeGate, SpeedGate, AGLAltGate
6.4.4 through 6.4.4.3	Envelop2
4.11.5.4	InAverage4Beam
4.11.4.2.2.2	InPhaseArrayBeam
4.11.2.3	SelectWaveform, AdjustWaveform
6.5.3.7.2.1	PositionClutterNotch, ApplyClutterNotch
6.5.3.7.2.2	AlarmSigProc
6.5.3.7.2.3	AlarmEclipse
6.5.3.7.2.3 through 6.5.3.5	SimpleSenseBeamLoss
6.5.3.1	CalcAntennaGain
4.11.5.4	InterpolateScanLoss
6.5.3.2.1	CallLAProp
6.5.3.2.2	Atmosphere
6.5.3.6	CalcSTCGain
6.4.5, Appendix b-3	TerrainMasked
6.5.5.1	FindCancelFactor
6.5.6 through 6.5.6.3.1	AlarmClutter
6.5.8.2	FluctuateRCS

A discussion of the Passive RF sensor can be found in Section 6.6 of the EADSIM Methodology Manual [2].

2.5.3 Functional Element Software Design

Calling Tree

The calling tree for the sensor functional element is shown in Figures 2.5-1, 2.5-2 and 2.5-3. It consists of the subtrees for the radar sensor and the passive RF sensor. The tree starts with the main program for the detection process and shows the first level of calls, including the procedure, Detect. The path from Detect follows SensorTgt and RadarTgt to Radar where the detailed computations for the EADSIM radar model are performed. Alternatively, the path from Detect to SensorTgt to RFSensorTgt to RFSensor leads to the procedures that perform computations for the passive RF sensor.

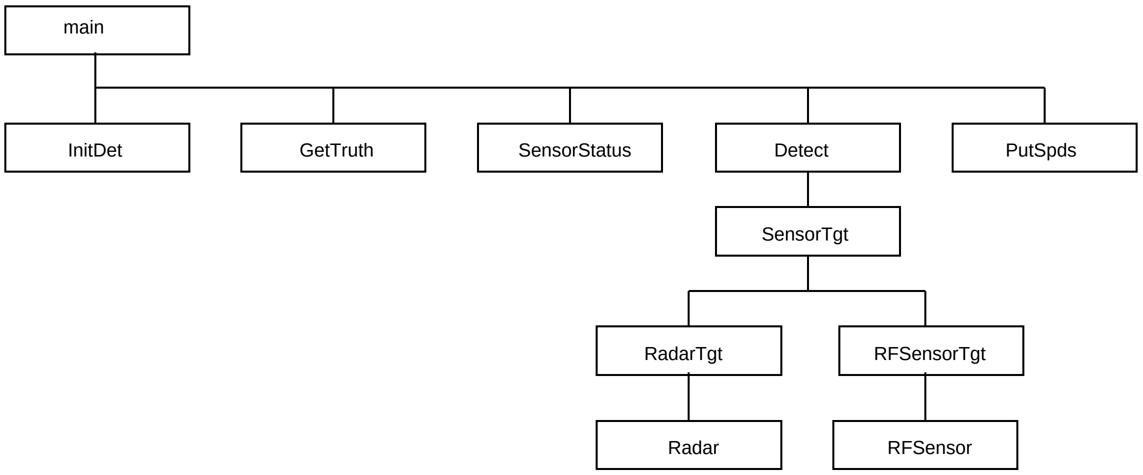

FIGURE 2.5-1. Calling Tree for Radar and RF Sensors.

Module Names and Descriptions for Radar Sensor

main - main is the driver for the detection model. It initiates the interprocess communication links. Scenario dependent data is input and the detection processing loop is performed.

InitDet - Initializes detection specific parameters for sensors and jammers.

GetTruth - GetTruth is used by sensor detection to get the truth data sent over the socket by flight processing.

SensorStatus - Sensor commands from C3I are received by this module for use in detection processing.

Detect - This module serves as the driver for all detection process including radar, infrared, signal intelligence, human intelligence, image intelligence, launch detection, passive RF, and radar warning receiver.

PutSpds - PutSpds is used to send detection status data back to C3I.

SensorTgt - This module loops through all of the available systems to determine which target systems can be seen by the sensors on the current platform. The presence of each sensor type is the checked and the appropriate sensor functions are called to continue the detection process.

RadarTgt - RadarTgt performs upper level processing for radar detection. Initialization of ALARM specific multipath and atmospheric parameters is performed. Target list search operations are performed to determine candidate radar targets. For candidate ABT and TBM targets, Radar is called to determine detection. Probabilistic detection is performed for candidate ground targets.

Radar - Computes signal-to-interference ratio for each target and determines detection based on a threshold level for deterministic detection or based on probability of false alarm and target fluctuation type for probabilistic detection.

RFSensorTgt - The module RFSensorTgt is the main control module in the detection process for the detection of radar, communications device, or jamming emissions by a passive sensor. The control module determines detections for all active passive sensors during a scenario interval. The methodology for determining the detection outcome will either be deterministically or probabilistically computed based on how the passive sensor is defined.

RFSensor - The module RFSensor handles preliminary filtering of targets in the passive sensor detection model. Targets are filtered based on range, active emitters, and AORs.

The module also handles logging of detection via the module LogSpds if the detection outcome is successful.

Radar Subtree

The subtree for the radar spans more than one page. The number at the end of one line indicates that the current level of calls is continued on the next line (which begins with the same number).

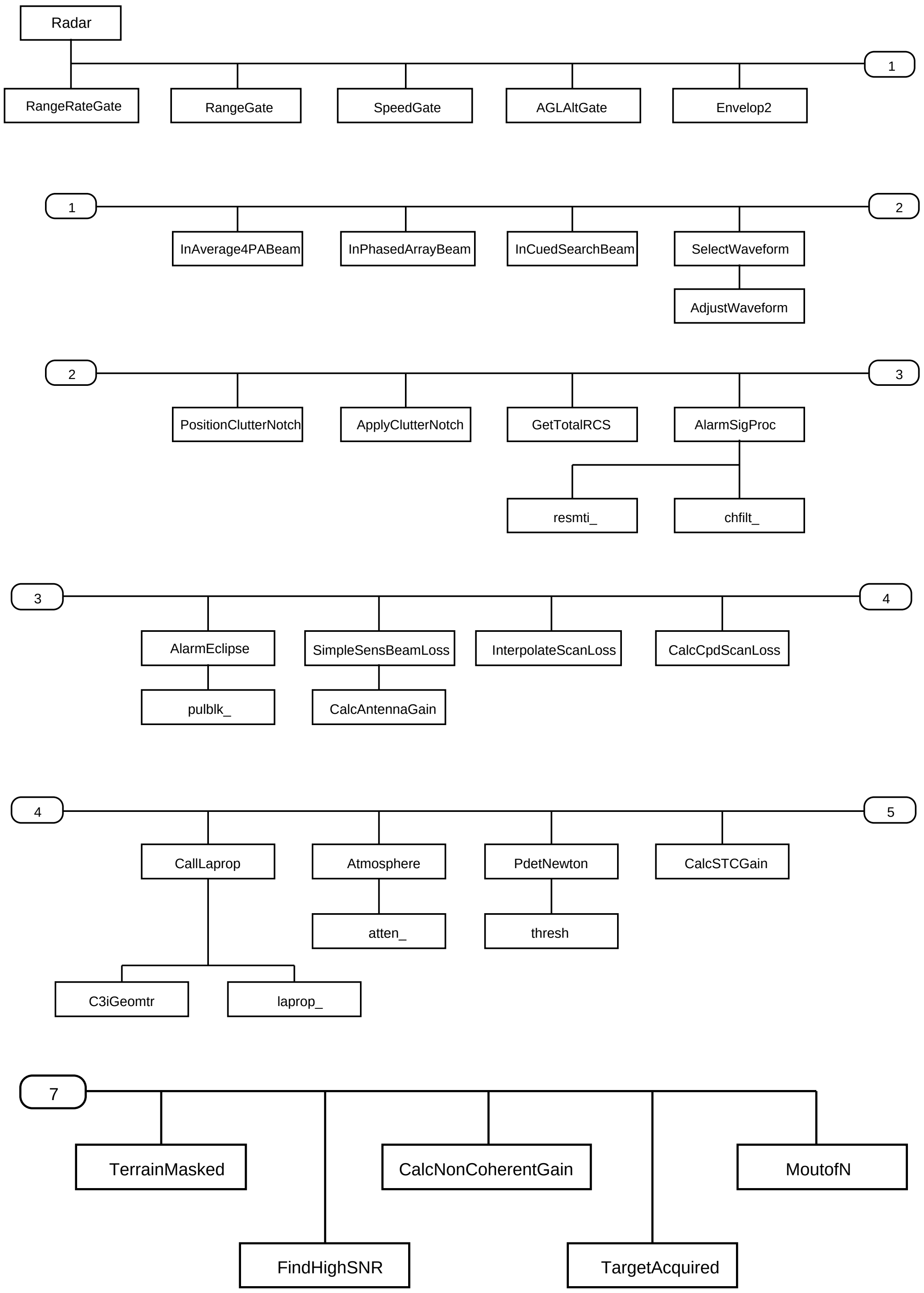

FIGURE 2.5-2. Radar Calling Tree.

Module Descriptions for Figure 2.5-2

RangeRateGate - RangeRateGate restricts detection to targets having relative velocity between user input minimum and maximum values.

RangeGate - RangeGate restricts detection to targets having ranges between user input minimum and maximum values.

SpeedGate - SpeedGate restricts detection to targets having absolute speeds between user input minimum and maximum values.

AGLAltGate - AGLAltGate restricts detection to targets having above ground levels between user input minimum and maximum values.

Envelop2 - Envelop2 restricts additional detection processing to targets within the sensor's defined field of view.

InAverage4PABeam - Determines if a target is within the partial coverage of a search sector of a compound sensor. The searching is assumed to be done by a four faced phased array radar with an unknown orientation and therefore an average beam shape is assumed.

InPhasedArrayBeam - Determines if a target is within the partial coverage of a search sector. The searching is assumed to be done by a phased array radar and therefore the target position is transformed to sine space.

InCuedSearchBeam - This module probabilistically determines whether or not a target in cued search is within the 3dB beamwidth of one of the beams searched during an interval

SelectWaveform - This function selects the appropriate waveform to maintain the specified SNR for managed dependent sensors. Note that noncoherent waveforms are treated as coherent waveforms for the purpose of selection.

AdjustWaveform - Determines if the waveform needs to be reduced because of eclipsing.

PositionClutterNotch - This module positions the clutter notch in aliased Doppler.

ApplyClutterNotch - This module determines whether the target is at a Doppler blindspeed.

GetTotalRCS - This module gets the RCS values from the target's RCS table.

AlarmSigProc - This function decides which signal processing routine to call. Once the decision is made preparation is made for the actual call to one of two routines -- the ALARM modules RESMTI or CHFILT.

resmti_ ALARM subroutine which calculates the response of the MTI system for a specific Doppler frequency.

chfilt_ ALARM subroutine which calculates the response of a Chebyshev filter.

AlarmEclipse - This function prepares to call the ALARM module PULBLK which is a FORTRAN module.

pulblk_ ALARM subroutine to find the eclipsing loss using staggered PRFs

SimpleSensBeamLoss - This module computes the off - boresight azimuth and elevation angles and the corresponding antenna gain for rectangular and circular beams

CalcAntennaGain - This executive antenna pattern function obtains the antenna gain and returns as follows:

1 for success in using detailed antenna pattern
0 for failure in using detailed antenna pattern

InterpolateScanLoss - This module performs linear interpolation for the computation of scan loss

CalcCpdScanLoss - Computes the average scan loss for a four faced phased array radar. It is assumed that the azimuth orientation of the antenna is unknown. For each elevation, the scan loss is averaged over - 45 to +45 degree azimuth angle.

CallLaprop - This function prepares to call laprop_(), which is the FORTRAN module from ALARM which handles multipath and diffraction. This module puts inputs required for CalcAntennaGain into an external structure (offbor), so that they can be accessible downstream in "c" without being visible in the ALARM FORTRAN routines themselves.

C3iGeomtr - This module sets up the geometry for the FORTRAN module LAPROP, which is the multipath module of ALARM30 that has been integrated into EADSIM.

laprop_ ALARM subroutine to compute multipath and diffraction.

Atmosphere - This function computes atmospheric attenuation by calling the ALARM FORTRAN routine atten. It returns the gain due to 1 - way atmospheric attenuation to the EADSIM C functions.

atten_ ALARM subroutine which computes atmospheric attenuation.

PdetNewton - Calculates the probability of detection using the Newton method.

thresh - ALARM subroutine to compute detection threshold based on Swerling fluctuation type and desired probability of detections and false alarm.

CalcSTCGain - This module applies the Sensitivity Time Control gain to the target signal.

TerrainMasked - Checks the line - of - sight between the given two positions and determines if the line of sight is masked by the terrain described by the given map or bald earth if no map specified.

FindHighSNR - This module finds the highest SNR in the SNR array.

CalcNonCoherentGain - This module computes the noncoherent processing gain.

TargetAcquired - This module determines whether or not the initial detection of a given target has already taken place and being tracked by the associated dependent sensor.

MoutofN - This function logs a detection result into a rotating window and determines if there have been enough detections (m) during a number of scans (n) corresponding to the size of the rotating window. This function is only used by sensors using the m out of n detection criteria.

Passive RF Sensor Subtree

Figure 2.5-3 shows the calling tree for the Passive RF sensor. The two branches at the second level, containing RFSensorProbDetect and RFSensorDetect represent the probabilistic and deterministic detection models.

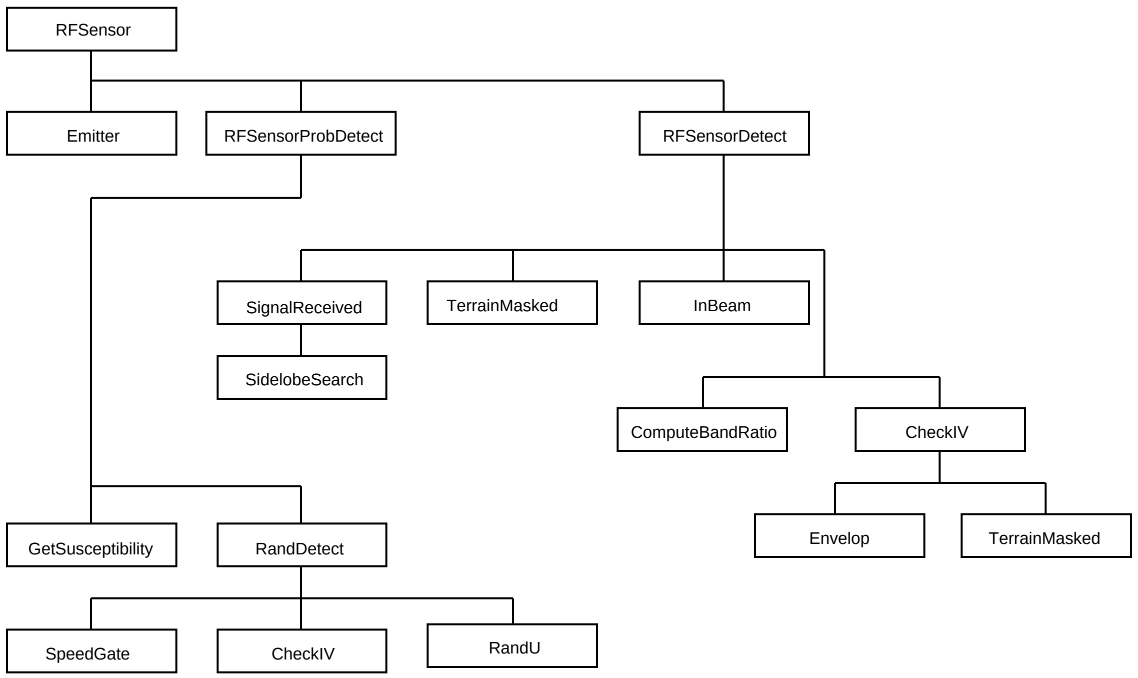

FIGURE 2.5-3. Passive RF Sensor Calling Tree.

Module Descriptions for Figure 2.5-3

Emitter - The module Emitter determines whether an target has any radar, jammer, or communications device emissions. If no emissions within the receiver band of the passive sensor are detected the target is not a candidate for detection. If the target has current emissions in the passive sensors receiver band then the target is eligible for detection by the passive sensor.

RFSensorProbDetect - The module RFSensorProbDetect is directly called by RFSensorTgt. This module computes detection outcomes for the passive sensor based on probabilistic methods. If the probability of detecting the emitter is greater than a normally distributed random number then the detection is successful. Otherwise the detection is flagged as a failure.

GetSusceptibility - The module GetSusceptibility determines the susceptibility of the target to being detected based on the type of sensor which is attempting to detect it. The susceptibility inputs are user defined at the System level for most sensor types. Sensor types which are not explicitly defined use the susceptibility numbers of the general class to which it belongs. There are no explicit susceptibility inputs for the passive sensor type.

However, there are susceptibility inputs for the Signal intelligence sensor type. The passive sensor was designed to mimic and improve on the SIGINT sensor model. As such the susceptibility inputs for the SIGINT sensor type apply to the passive sensor as well.

The use of the system/sensor based susceptibility is limited to the probabilistic detection methodology.

RandDetect - This module uses a probability of detection to determine if the input sensor can see the input target.

SpeedGate - Filters out targets whose speeds fall outside a speed window

CheckIV - This function checks the LOS between two systems to determine if it is unobstructed by terrain. If the target is viewable to the sensor, then the function returns spherical and rectangular position coordinates of the target relative to the sensor. Otherwise the failure flag is returned.

RandU - Generate a uniformly distributed random number using the linear congruential method.

RFSensorDetect - The module RFSensorDetect is directly called by RFSensorTgt. This module computes detection outcomes for the passive sensor based on the RF signal emitting from the target present at the passive sensor antenna. If the received signal is above a user defined threshold the detection is successful. Otherwise the detection is flagged as a failure.

SignalReceived This function determines if the Radar Warning Receiver Sensor succeeds or fails to receive a signal from the Target Sensor.

SidelobeSearch - This module determines if the signal received from a target sensor is detectable.

TerrainMasked - InBeam - This function takes information about a pair of antennas and determines whether the target antenna is in beamwidth of the vertex antenna. It works for antennas pointed to a specific location or pointed in a direction to the heading angle of the platform to which it is attached, as would sometimes be the case with a jammer. The ‘vertex’ system is the system that generates an ellipsoid or rectangular volume. This volume may or may not contain the ‘target’ system.

InBeam - This function takes information about a pair of antennas and determines whether the target antenna is in the beamwidth of the vertex antenna. It works for antennas either pointed to a specific location or pointed in a direction relative to the heading angle of the platform to which it is attached, as would sometimes be the case with a jammer. The 'vertex' system is the system that generates an ellipsoid or volume. This volume may or may not contain the 'target' system.

ComputeBandRatio - The module ComputeBandRatio determines the bandwidth overlap between the passive sensor and the emitter it is attempting to detect. The bandwidth overlap ratio is used to adjust the signal power at the passive sensor antenna when an communication device or jammer is being evaluated. This signal power adjustment accounts for the communication device or jammer power being averaged over the entire operating frequency therefore reducing the power coming from any one channel to an average power level.

Envelop - Determines if a target platform is within the envelope of the sensor. This is designed for use with probabilistic type sensors.

2.5.3.1 Module Descriptions and Logic Diagrams Radar

The Detection process module, Radar, performs detection processing for sensors of type radar. Inputs and outputs for the module, Radar, are listed in Table 2.5-1 and the steps performed by this module are shown in Figures 2.5-4 through 2.5-10. Modules which are not used for a Flexible SAM radar are not shown. In particular, the modules dealing with helicopter detection and airborne pulse Doppler radars have been omitted.

Due the large size of the logic diagram for this module, 3 breakouts, A1, A2, and A3 have been made. Breakout A1 covers the logic used to perform FOV checks and algorithms used to determine if the target is in the current beam position of a compound sensor. Breakout A2 shows the flow for the probabilistic detection portion of the module. Breakout A3 shows the flow for the deterministic detection portion of Radar. The block descriptions that follow each of the diagrams follow the block numbers in the respective diagram.

TABLE 2.5-1. Inputs and Outputs for Radar.

Inputs:	
Missile	Pointer to targeted missile
ABT	Pointer to targeted air breathing threat
Sensor	Pointer to the sensor doing the viewing.
FOVLimit	An array of field of view limits whose contents depend on
The FOV type:	
RECTANGULR	
FOVLimit[0]	Start azimuth.
FOVLimit[1]	Stop azimuth.
FOVLimit[2]	Start elevation.
FOVLimit[3]	Stop elevation.
CIRCULAR	FOVLimit elements 0,1,2 contains the components of the boresight vector from which the maximum off boresight angle is calculated.
RotBF	Matrix which when multiplied by the sensor to target vector, will rotate the vector to the sensor body frame.
RotRF	Matrix which when multiplied by the sensor to target vector, will rotate the vector to the sensor radar face.
Scenario	Pointer to current scenario.
AngPos	Pointer to the Az, El, Range from the sensor to tgt in BF coord.

TABLE 2.5-1. Inputs and Outputs for Radar. (Contd.)

Inputs:	
SystemJammers	List of jammers that could effect this sensor
SenSysNum	ID number of the current sensor.
TargetIndex	Index into dependent sensor target list
Outputs:	
rburn	Pointer to maximum burnthrough range for the radar.
SNR	Pointer to the final signal-to-interference ratio.

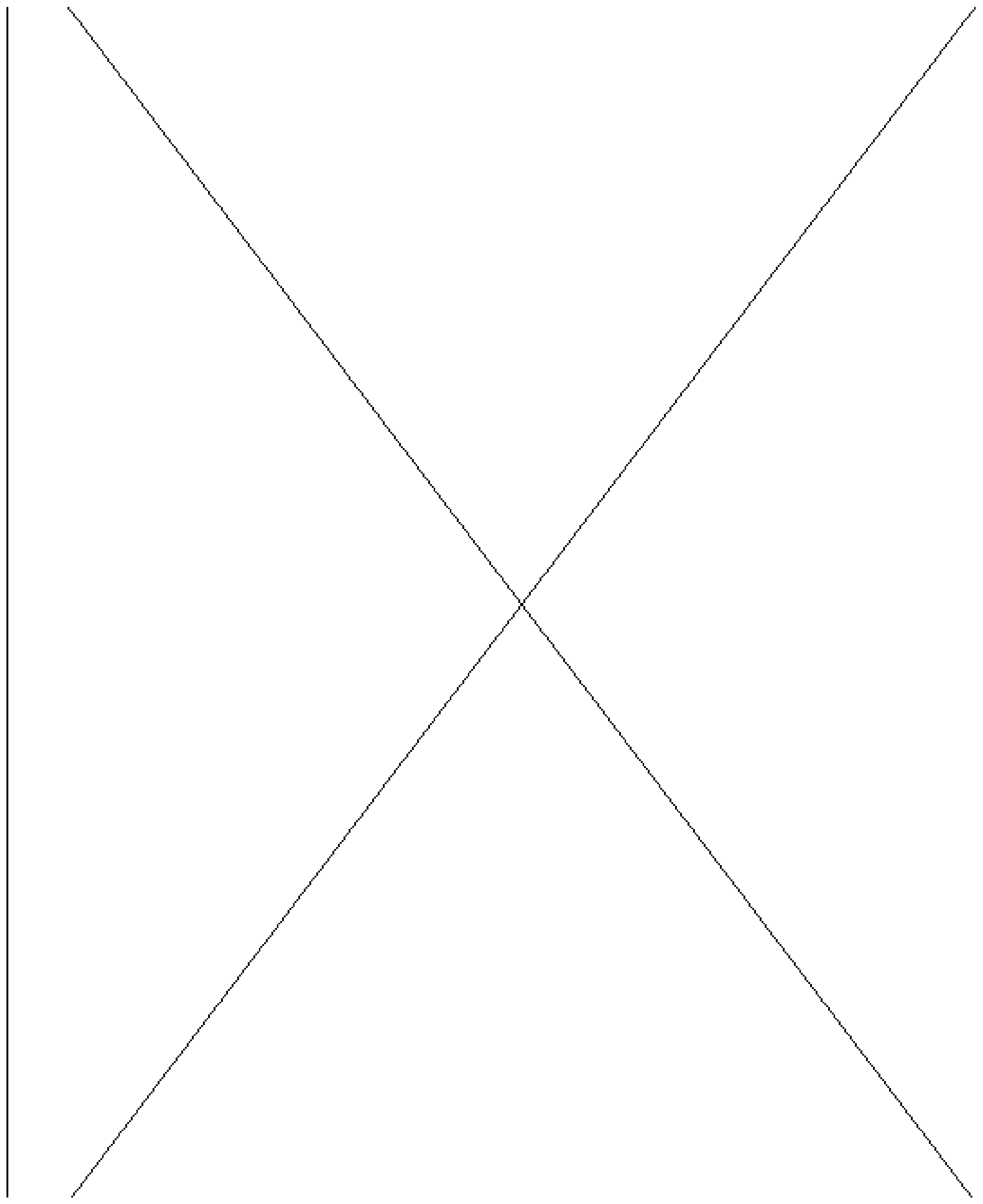

FIGURE 2.5-4. Radar Sensor Processing Logic.

Block 1: If the radar is compound, a search is performed for the appropriate sensor substructure. The result is the pointer to the current compound sensor element structure.

Block 2: The Field of View (FOV) type determines the types of gating logic available for limiting the detection process. FOV NONE is provided primarily to allow the user to set up sensor parameters and determine detection characteristics without the limitations imposed by FOV, intervisibility, and gating logic among others. The call to RangeRateGate is used only to compute the range rate for use later in Radar. The selection of FOV NONE sets a flag that prevents the application of the range rate gating logic. When a rectangular or circular FOV is specified, all 4 gate limitations are available.

Block 3: In this module, the range rate is always calculated. If the flag for this routine is set and the range rate is outside the user set limits, the detection flag is set to indicate a detection failure due to range rate.

Block 4: Range gating does not have a user controlled flag and is thus always applied when called. If the target range is outside the user input limits, the detection flag is set to indicate detection failure due to range.

Block 5: Limits on the target speed is applied only if the user input flag for this routine is set. If the target speed is outside the user input limits, the detection flag is set to indicate detection failure due to speed.

Block 6: Limits on the target altitude above ground level are applied only if the user input flag for this routine is set. If the target altitude above ground level is outside the user input limits, the detection flag is set to indicate detection failure due to altitude.

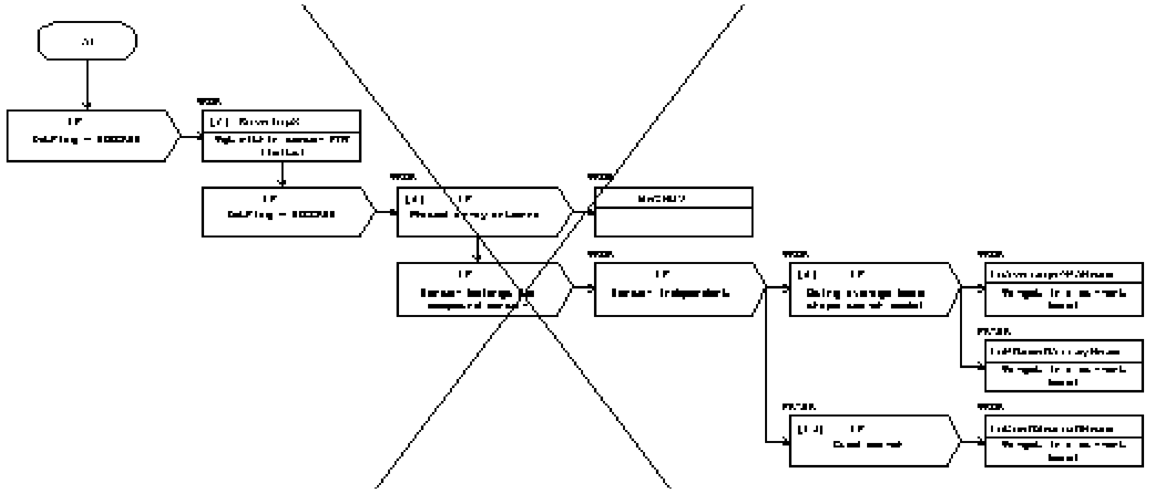

FIGURE 2.5-5. Breakout A1 FOV Processing.

Block 7: If the previous tests have not changed the detection flag to the failure condition, Envelop2 is called. If the FOV is rectangular the target azimuth and elevation is tested against azimuth and elevation FOV limits. If the FOV is circular, the target angle with respect to the FOV center is tested against a maximum angle limit. The function return value sets the detection flag to either a success condition or a indication of the type of FOV failure.

Block 8: If the radar has a phased array antenna, GetRUV is called to compute the U, U dot, V, and V dot in the RUV coordinate system.

Block 9: If the sensor is part of a independent, compound sensor, this test determines the model to be used to determine if the target is within the current search sector. If the average beam shape search model is selected, InAverage4PABeam determines if a target is within the partial coverage of a search sector. The searching is assumed to be done by a four faced phased array radar with an unknown orientation and therefore an average beam shape is assumed. The target position is left in real space. Otherwise InPhasedArray determines if a target is within the partial coverage of a search sector. The searching is assumed to be done by a phased array radar and therefore the target position is transformed to sine space.

Block 10: If the sensor is part of a dependent, compound sensor, this test determines is this is a cued search. If it is, InCuedSearchBeam is called to probabilistically determine whether or not the target in cued search is within the 3 dB beamwidth of one of the beams searched during the current interval.

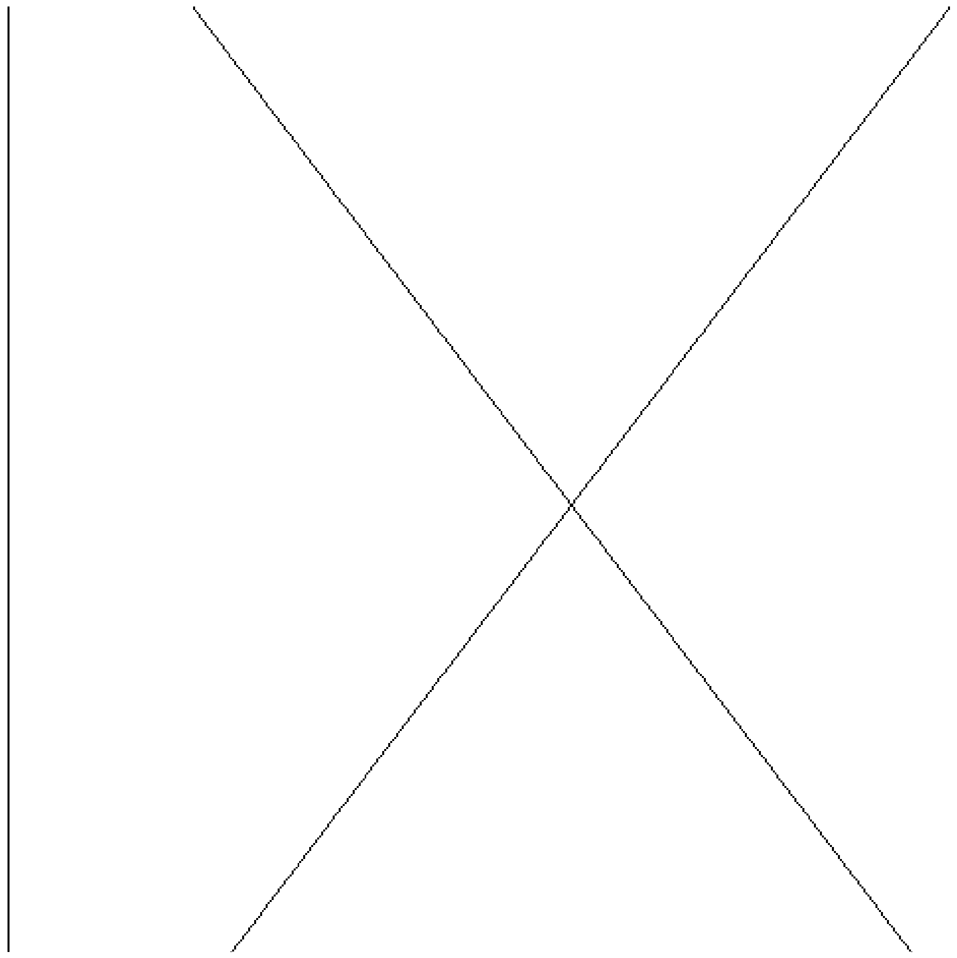

FIGURE 2.5-6. Radar Sensor Processing Logic.

Block 11: This block of code determines whether the radar constant K0 computed for a single pulse of a compound radar waveform needs to be adjusted due to a change in pulse length. If the sensor is dependent and is in a cued search mode, the maximum waveform pulse length is used along with it's corresponding K0, else SelectWaveform is called to select the correct number of pulses and pulse length to achieve the desired minimum signal-to-noise ratio. The correct value of K0 is also set. If the sensor is independent, K0 is set based upon the current pulse length.

Block 12: When a coherent waveform is used, the target Doppler frequency is always computed. Then if the simple coherent processing option is selected, PositionClutterNotch is called to determine the location of the clutter notch in the aliased Doppler domain. The center of the clutter notch is positioned to match the Doppler frequency of the ground directly under the radar. For a stationary radar, the clutter notch is thus centered about zero frequency. The width of the clutter notch is determined by user input. The response within the notch is identically equal to zero. ApplyClutterNotch is then called to determine if the aliased target Doppler frequency is within the clutter notch. If it is, the detection flag is set to indicate failure due to blindspeed.

Block 13: If the detection is not set to failure due to blindspeed, GetTotalRCS is called to obtain the total RCS for the current target. The total RCS is computed from the RCS values for the components of the target and can either be a user specified constant value or read from input tables.

Block 14: This block is checked if coherent processing is selected and the target RCS fluctuation model is not Swerling II or Swerling IV. The assumption is that these two models imply rapid fluctuation between coherent pulses and thus no coherent processing gain is achieved. If the detailed coherent model (ALARM) is selected, AlarmSigProc is called to compute the coherent processing gain using ALARM routines for either MTI or pulse Doppler. Otherwise, the simple coherent processing gain is computed.

Block 15: When a radar is using noncoherent pulse integration on a Swerling I or Swerling III target, a RCS fade will cause a loss in detection because, by definition, the RCS is constant over the pulse integration interval. By use of different frequencies for each pulse, the target will behave more like a Swerling II or Swerling IV which results in a higher probability of detection. If the frequency hop option is selected, Swerling I is changed to Swerling II, and Swerling III is changed to Swerling IV. Note that both coherent and noncoherent integration can be used on the same radar waveform. The coherent integration is applied to the specified single pulse. The output of the coherent integration is then treated as the input pulse for the noncoherent integration.

Block 16: If coherent processing is used, AlarmEclipse is called to compute the processing loss due to eclipsing and transmitter pulse blanking. AlarmEclipse acts as initialization and an interface to pulblk_ which is an ALARM routine.

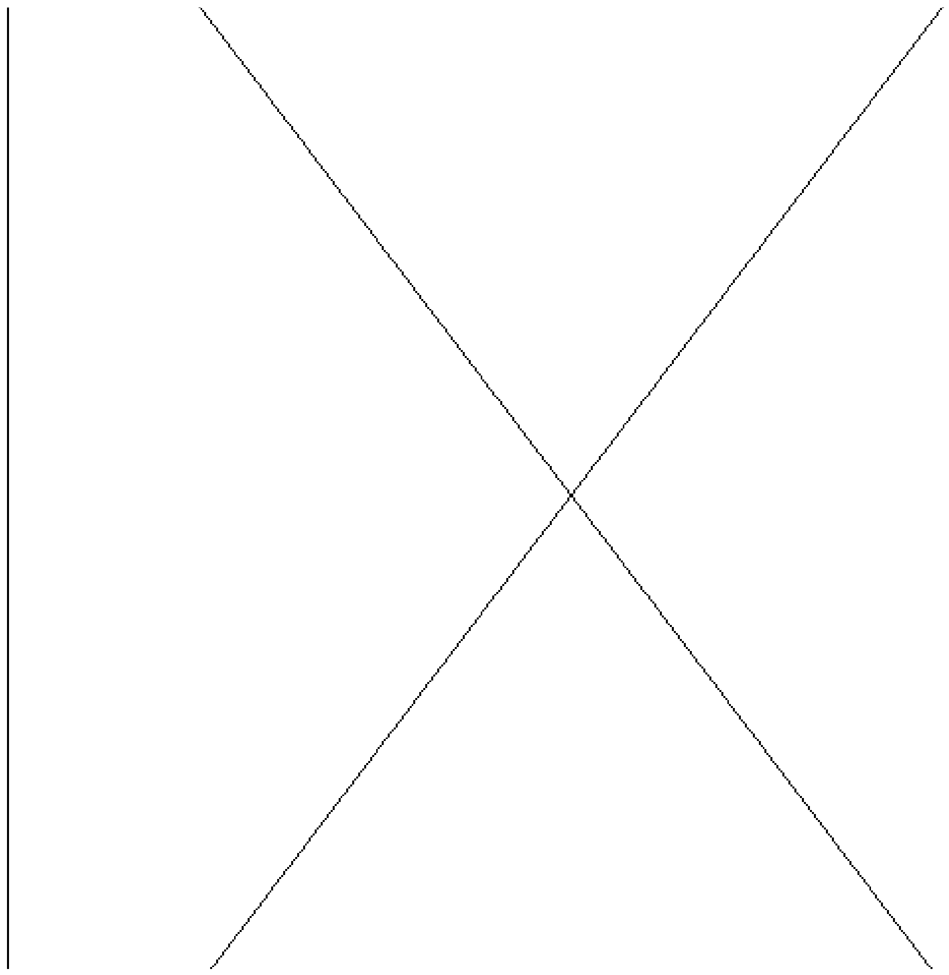

FIGURE 2.5-7. Radar Sensor Processing Logic.

Block 17: If the detection flag is still set to success, SimpleSensBeamLoss is called to compute the effect of the sensor antenna pattern on target SNR. If a scan raster pattern is used, the mainlobe will not be centered on the target. The minimum angular offset during the scan is determined and the corresponding gain is returned. If the user has selected the average scalloping loss option, the corresponding loss is returned. If the sensor is simple phased array either the computed off-normal scan loss or an optional average scan loss will be returned. The average scan loss assumes random target position within a phased array scan pattern and computes the average loss as a function of the relative beam spacing.

Block 18: This block calculates scan losses when a compound sensor is used. If the average beam shape search model is selected, InterpolateScanLoss is called. This routine assumes a 4 face phased radar for which the azimuth orientation is unknown. The user inputs the phased array layback angle. At program initialization, a numeric integration is performed to create a table of average loss versus target elevation angle. In runtime, the table is interpolated to obtain the computed loss. Otherwise, CalcCpdScanLoss is called to compute the cosine of the angle between the array normal and the target which is used in Radar to compute the loss.

Block 19: If multipath is selected, CallLaprop is called. It acts as initialization and an interface to the ALARM FORTRAN routine laprop_. The ALARM routines have been modified to use the EADSIM antenna patterns.

Block 20: If atmospheric loss is selected, Atmosphere is called. It acts as initialization and an interface to the ALARM FORTRAN routine atten_.

Block 21: The user can choose between two type of detection: deterministic and probabilistic. Deterministic detection computes the SNR based upon the target RCS, range, and radar performance characteristics. The SNR is then compared to a user input threshold level. If the SNR is below the threshold, the detection flag is set to failure and no additional detection processing is attempted. If the threshold is exceeded, degrading effects such as jamming and clutter are computed to determine if the SNR drops below the threshold level. The threshold level is checked as often as possible to permit an exit once the SNR drops below the threshold in order to conserve computation. The probabilistic detection checks for detection only at the end of all of the calculations because the probability of detection depends only on the final SNR. For this reason, probabilistic detection will be slower than deterministic especially when effects such jamming and clutter are included.

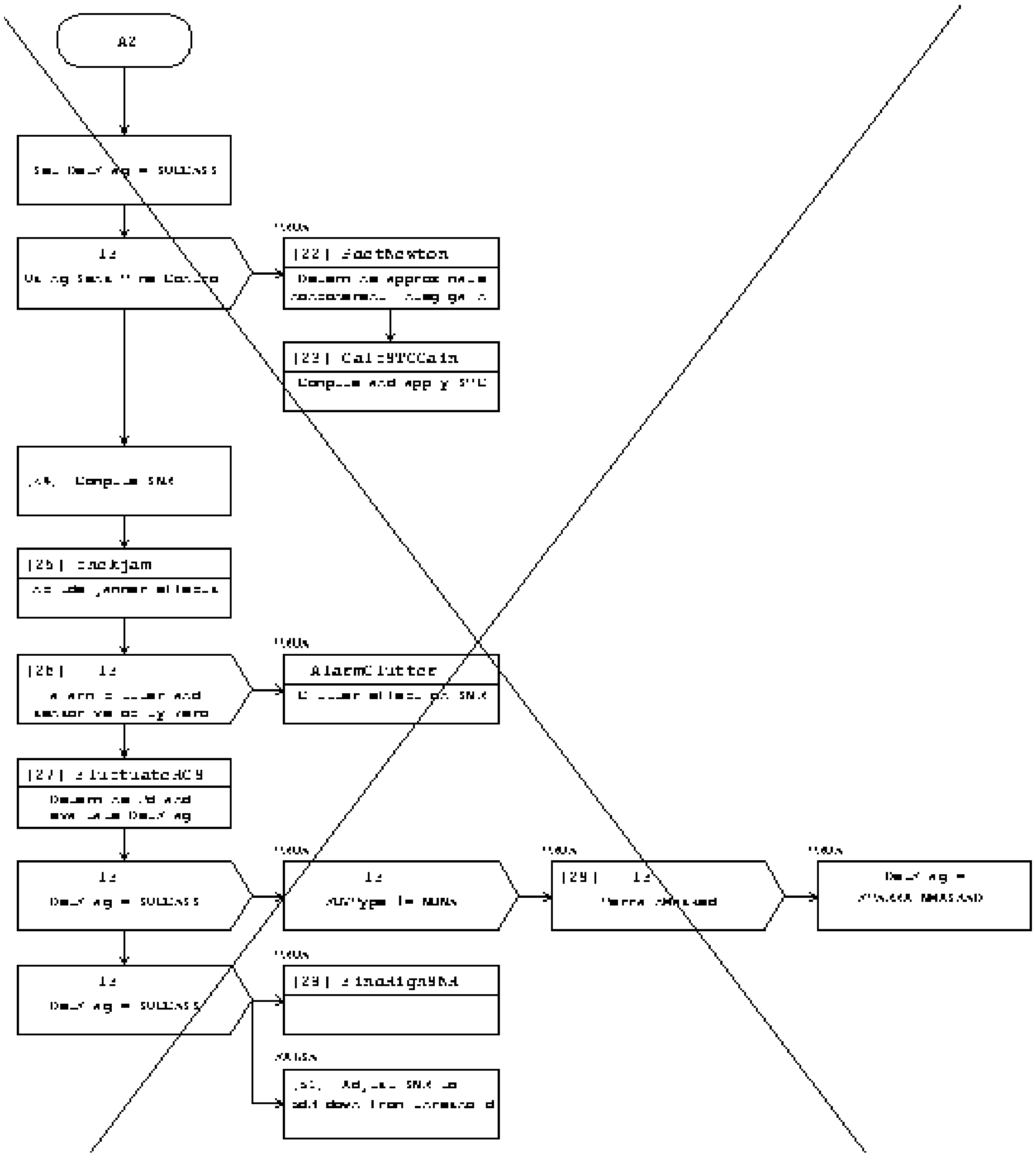

FIGURE 2.5-8. Breakout A2 Probabilistic Detection Processing.

Block 22: If sensitivity-time-control (STC) is selected and noncoherent integration is used, PdetNewton is called to determine the approximate noncoherent integration gain. It is required because CalcSTCGain operates on the basis of the specification of the asymptotic gain for a single pulse at zero range.

Block 23: CalcSTCGain models sensitivity-time-control (STC). The user can select a characteristic having zero, one, or two regions in range or altitude which have a specified offset from the nominal STC characteristic which varies as R to the fourth power. The gain scaling is specified by input of the asymptotic SNR at zero range on a 1 square meter target for a single pulse of the waveform.

Block 24: In this block, the peak SNR based upon ideal conditions is calculated. The target RCS, radar constant (for a single pulse), coherent integration gain, atmospheric loss, multipath, and eclipsing factors are included in the computation.

Block 25: This module computes the degradation of SNR due noise jammers. The received jammer power is scaled by the overlap ratio of the jammers and the sensor, and the sensor antenna gain in the jammer direction. If multipath is selected for the sensor, it is applied for the path between the jammer and the sensor (one-way gain). The same is true for atmospheric attenuation.

Block 26: If ALARM clutter generation is selected and the sensor is stationary, AlarmClutter is called. This routine serves as initialization and an interface to the ALARM FORTRAN routines clutpd_ and clutpu_. which compute ground clutter for pulse Doppler and MTI waveforms. A third routine, clutnc_, is a modified version of clutpu_ which is used to compute the ground clutter for noncoherent radars.

Block 27: FluctuateRCS computes the probability of detection, and then evaluates the detection by making a random draw from a uniform distribution and comparing it to the probability of detection. The algorithm for the optional adaptive radar (see Block 32) is contained within this module. As part of the adaptive radar computation, PdetNewton is called if the radar performs noncoherent integration so that a estimate of the noncoherent integration can be used in the adaptive computation. Once the final SNR is determined, PdetNewton is called to determine the probability of detection.

Block 28: If detection is still successful and the FOV is either rectangular or circular, TerrainMasked is called to determine whether there is line of sight between the sensor and the target. If not, the detection has failed. Not shown on the figure is an additional check on whether multipath is used. The multipath calculation in the ALARM code includes a check for line of sight. Thus TerrainMasked is not exercised when multipath is active.

Block 29: When the ALARM pulse Doppler processing is active, it is necessary to keep track of the SNR for each of the 1 to 4 PRFs specified by the user. FindHighSNR determines which of SNR values are the largest for use by the detection processing.

Block 30: The detection has failed. The returned SNR is adjusted to be 3 dB below the threshold. This value is used in C3I for radars which can adjust the pulse length and/or the number of noncoherently integrated pulses. Placing the SNR 3 dB below the threshold level gives the adjustment a convenient starting point for the subsequent pulse adjustments.

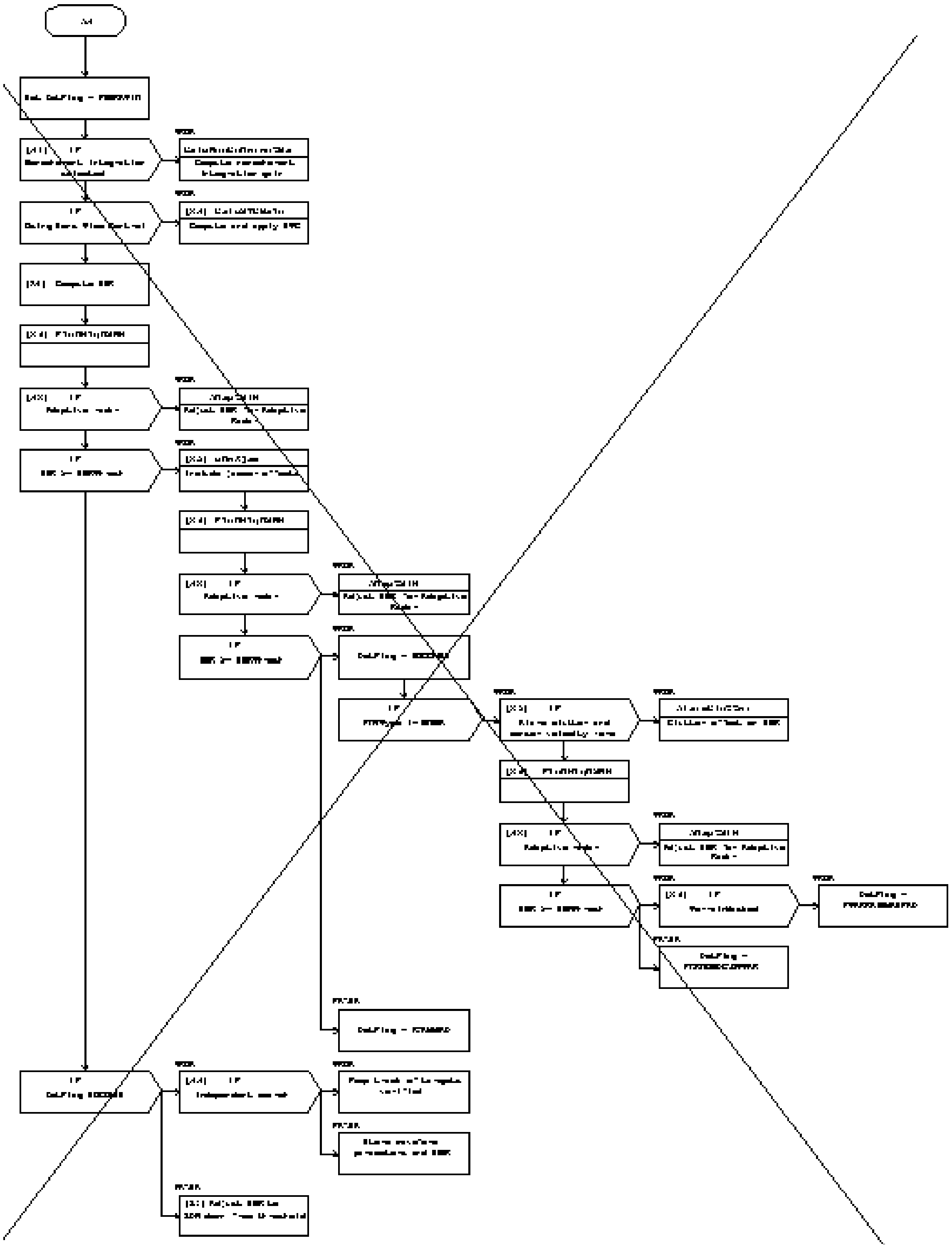

FIGURE 2.5-9. Breakout A3 Determination Detection Processing.

Block 31: When deterministic detection is selected, CalcNonCoherentGain is used to compute an approximation to the noncoherent integration gain. Curve fits to data generated for a 0.5 probability of detection and a 1.E-06 probability of false alarm are used.

Block 32: When the adaptive radar option is selected, AdapSIR is called to adjust the SNR. If the input SNR is below the threshold level, it is adjusted up to the threshold level unless the adjustment exceeds a user input maximum value in which case the adjust equals the maximum.

Block 33: If the detection is a success and this is an independent search, a additional check is made to see if this is a verify pulse. If so, the counter for the number of targets verified is incremented.

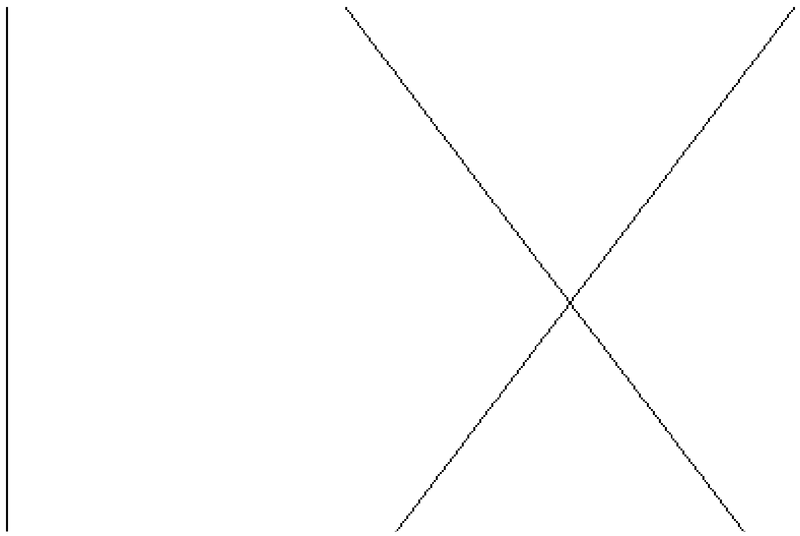

FIGURE 2.5-10. Radar Sensor Processing Logic.

Block 34: If the user has selected the m out of n option, MoutofN is called. In this routine, this and the previous m-1 sensor looks are scanned to see if a least n produced “hits”. If not, the detection flag is set to failure. Thus the calling routine treats the m out of n result as the detection report instead of the detection (or lack thereof) from the current look.

Block 35: If a dependent sensor gets a detection, the cued search flag (if it is set) is cleared thus ending the cued search.

Module Descriptions and Logic Diagrams for RFSensorTgt

RFSensorTgt - The module RFSensorTgt is the main control module in the detection process for the detection of radar, communications device, or jamming emissions by a passive sensor. The control module determines detections for all active passive sensors during a scenario interval. The methodology for determining the detection outcome will either be deterministically or probabilistically computed based on how the passive sensor is defined. Inputs and outputs for RFSensorTgt are listed in Table 2.5-2 and the steps performed by the module are shown in Figure 2.5-11.

TABLE 2.5-2. Inputs and Outputs for RFSensorTgt.

Inputs:	
Scenario	Pointer to Scenario data
SenSysNum	ID of Searching Platform
Sensor	Pointer to detecting sensor data
FOVLimit	Field of view data
RotBF	Body Frame rotation matrix
List	Pointer to sublists containing target data
Outputs:	
DetFlag	Flag indicating detection status
Count	Array of success/fail categories

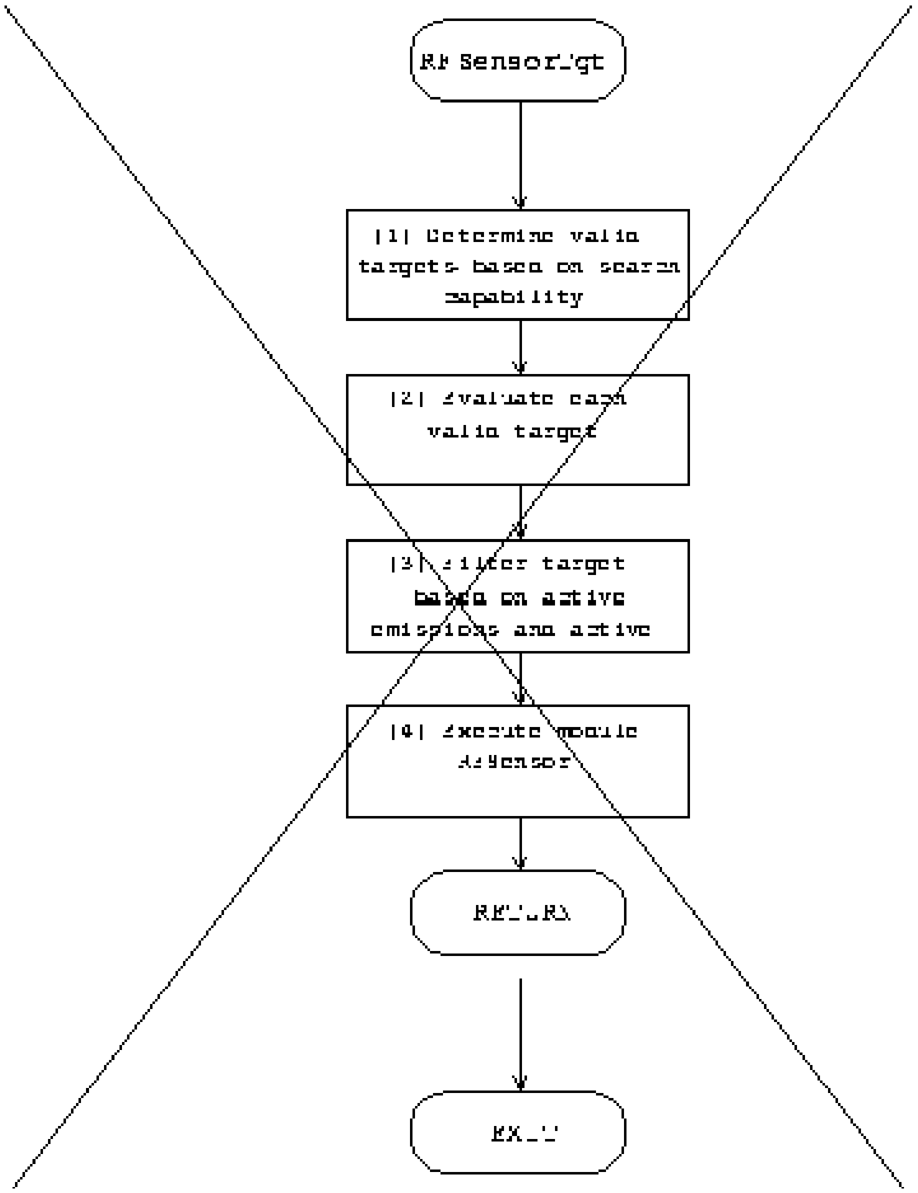
RFSensorTgt
[1] Determine valid targets based on search capability
[2] Evaluate each valid target
[3] Filter target based on active emissions and active
[4] Execute module RFSensor
RETURN
EXIT

FIGURE 2.5-11. RFSensorTgt Processing Logic.

__**Block 1.**__ Determine the valid targets for the sensor based on search capabilities

__**Block 2.**__ If the sensor is dependent search loop over the cued target list. Otherwise loop all targets defined in the scenario.

__**Block 3.**__ Filter targets based on preliminary tests. TBMs are excluded from being detected by the passive sensor. For non TBM targets the target must be active and emitting RF signals within the passive sensors operating frequency.

__**Block 4.**__ Determine the detection outcome.

RFSensor - The module RFSensor handles preliminary filtering of targets in the passive sensor detection model. Targets are filtered based on range, active emitters, and AORs. The module also handles logging of detection via the module LogSpds if the detection outcome is successful. Inputs and outputs for RFSensor are listed in Table 2.5-3 and the steps performed by the module are illustrated in Figure 2.5-12.

TABLE 2.5-3. Inputs and Outputs for RFSensor.

Inputs:	
Scenario	Pointer to Scenario data
SenSysNum	ID of Searching Platform
Sensor	Pointer to detecting sensor data
Target	Pointer to target platform data
FOVLimit	Field of view data
RotBF	Body Frame rotation matrix
Outputs:	
DetFlag	Flag indicating detection status
Count	Array of success/fail categories

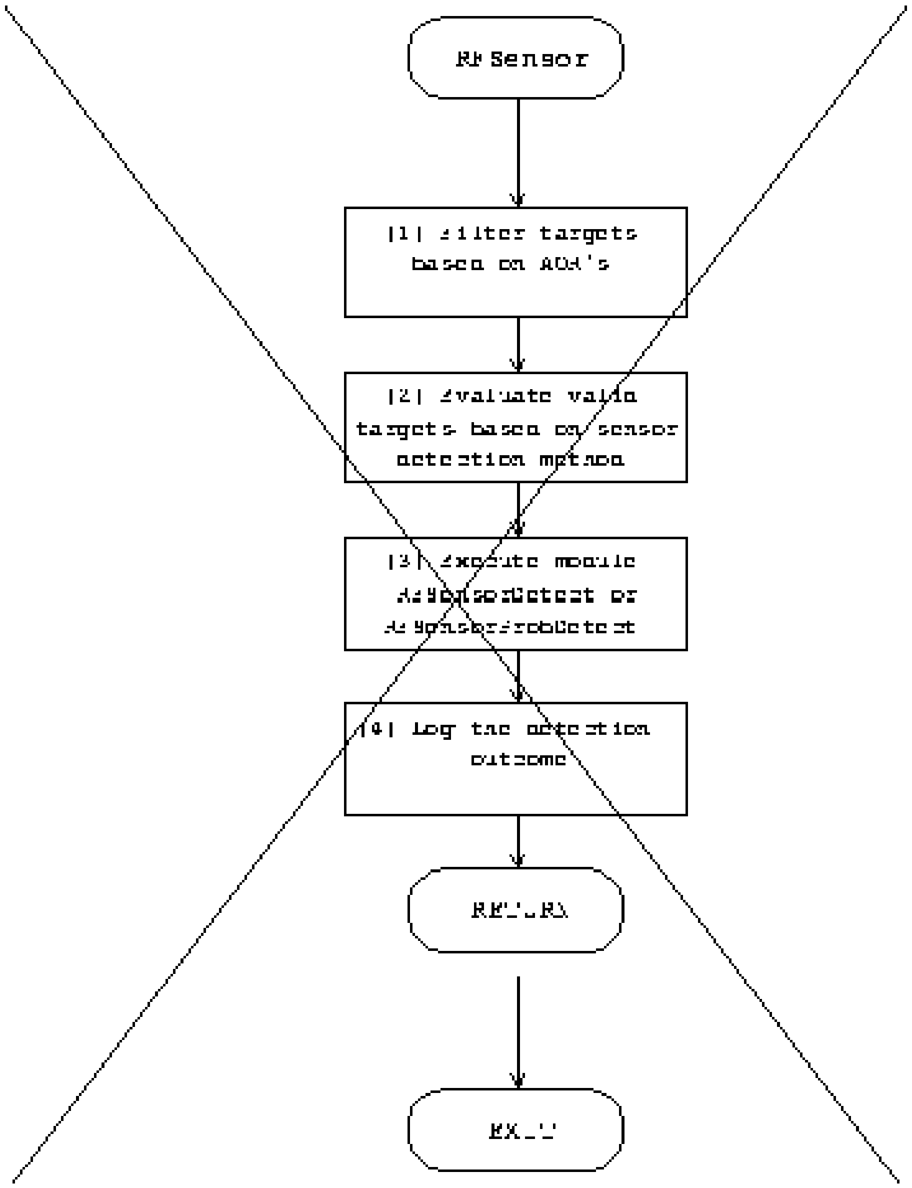
RFSensor
[1] Filter targets based on AOR's
[2] Evaluate valid targets based on sensor detection method
[3] Execute module RFSensorDetect or RFSensorProbDetect
[4] Log the detection outcome
RETURN
EXIT

FIGURE 2.5-12. RFSensor Processing Logic.

Block 1. If filtering targets based on area of responsibility, determine whether the target is within any AORs associated with the host platform. If not the detection outcome is logged as a failure. Otherwise the detection evaluation proceeds to the next item.

Block 2. If the target does not have detectable emitters log the detection outcome as a failure. Otherwise, determine the detection outcome based on either the probabilistic or deterministic method.

Block 3. If the detection was successful log the information via LogSpds.

RFSensorDetect - The module RFSensorDetect is directly called by RFSensorTgt. This module computes detection outcomes for the passive sensor based on the RF signal received at the passive sensor antenna. If the received signal is above a user defined threshold the detection is successful. Otherwise the detection is flagged as a failure. Inputs and outputs for RFSensorDetect are listed in Table 2.5-4 and the steps performed by the module are illustrated in Figure 2.5-13.

TABLE 2.5-4. Inputs and Outputs for RFSensorDetect.

Inputs:	
TargetOpFac	Pointer to target platform data
SenFreqLo	Sensor low frequency
SenFreqHi	Sensor high frequency
Sensor	Pointer to detecting sensor data
FOVLimit	Field of view data
RotBF	Body frame rotation matrix
AngPos	Target azimuth and elevation info
Scenario	Pointer to scenario data
Outputs:	
SpdsFlag	Flag containing info detected emitter type

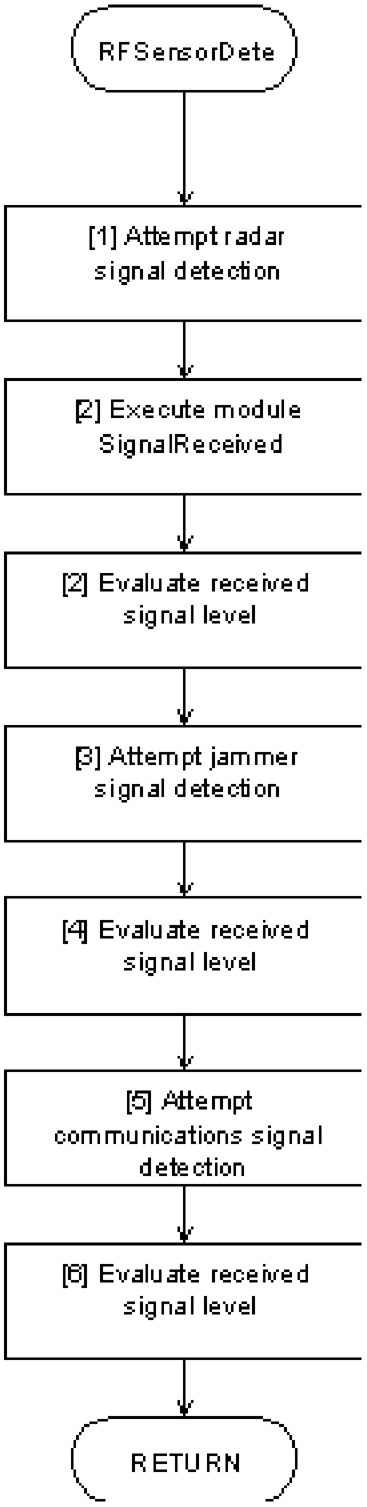
RFSensorDete
[1] Attempt radar signal detection
[2] Execute module SignalReceived
[2] Evaluate received signal level
[3] Attempt jammer signal detection
[4] Evaluate received signal level
[5] Attempt communications signal detection
[6] Evaluate received signal level
RETURN

FIGURE 2.5-13. RFSensorDetect Processing Logic.

Block 1. Attempt to detect radar signals from the target. If the radar is within the operating frequency of the passive sensor then check field of view and terrain masking.

Block 2. Determine the signal power received from the radar. If the signal level is adequate and the signal was determined to be intercepted then flag the detection as successful.

Block 3. Attempt to detect jammer signals from the target if no radar signals were intercepted. If the jammer is active and within the operating band of the passive sensor it is a valid candidate for detection. Check field of view constraints and terrain masking.

Block 4. Determine the signal power received from the jammer. If the signal level is adequate and the signal was determined to be intercepted then flag the detection as successful.

Block 5. Attempt to detect communications signals from the target if no radar or jammer signals were intercepted. If the communications device is active and within the operating band of the passive sensor it is a valid candidate for detection. Check field of view constraints and terrain masking.

Block 6. Determine the signal power received from the communications device. If the signal level is adequate and the signal was determined to be intercepted then flag the detection as successful.

RFSensorProbDetect - The module RFSensorProbDetect is directly called by RFSensorTgt. This module computes detection outcomes for the passive sensor based on probabilistic methods. If the probability of detecting the emitter is greater than a normally distributed random number then the detection is successful. Otherwise the detection is flagged as a failure. Inputs and outputs for RFSensorProbDetect are listed in Table 2.5-5 and the steps performed by the module are illustrated in Figure 2.5-14.

TABLE 2.5-5. Inputs and Outputs for RFSensorProbDetect.

Inputs:	
TargetOpFac	Pointer to target platform
SenFreqLo	Sensor low frequency
SenFreqHi	Sensor high frequency
Sensor	Pointer to detecting sensor data
FOVLimit	Field of view limit data
RotBF	Body frame rotation matrix
AngPos	Target azimuth and elevation data
Scenario	Pointer to scenario data
Outputs:	
SpdsFlag	Flag indicating detected emitter type
DetFlag	Flag indicating detection status

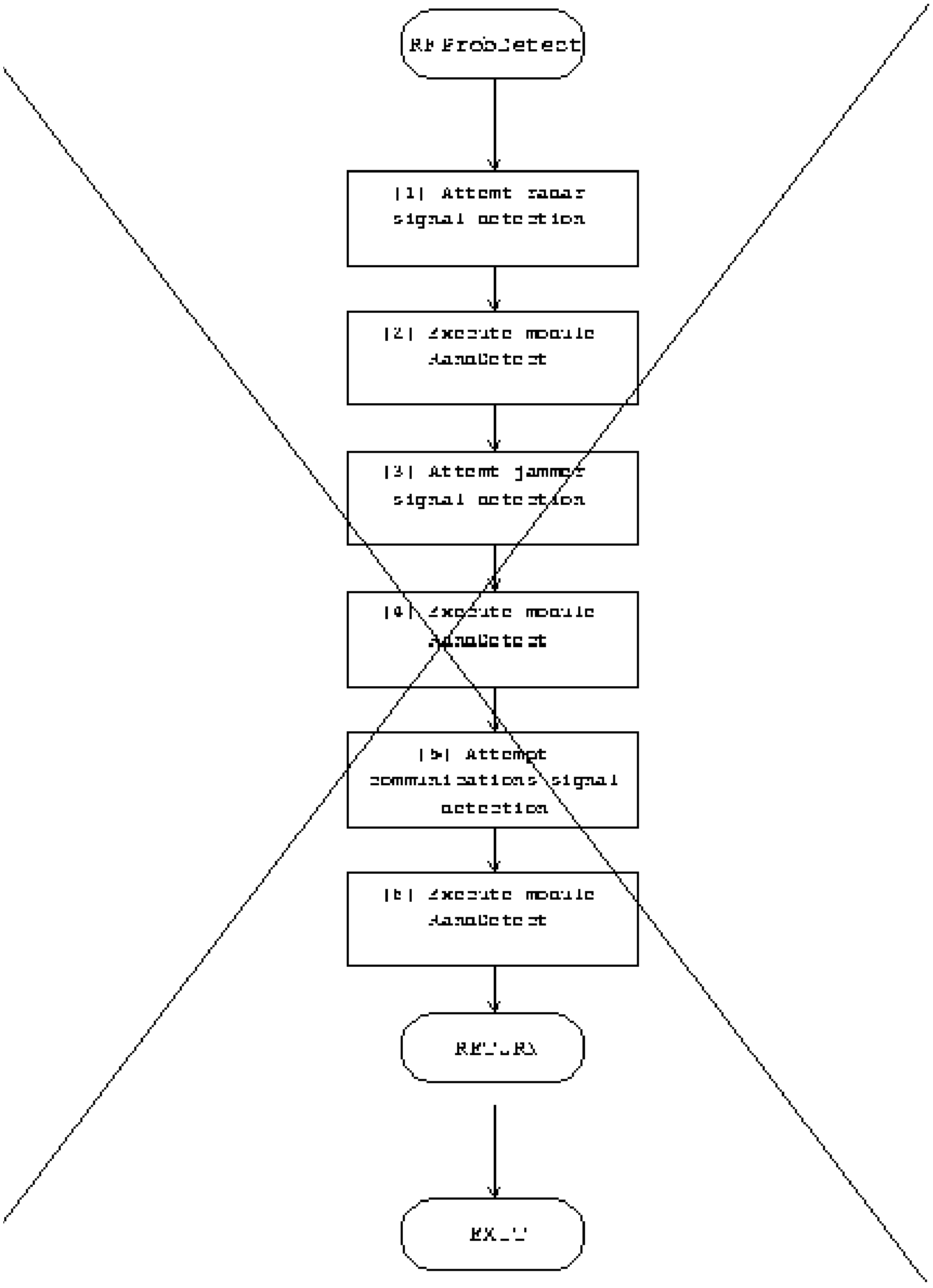
RF ProbDetect
[1] Attemt radar signal detection
[2] Execute module RandDetect
[3] Attemt jammer signal detection
[4] Execute module RandDetect
[5] Attempt communications signal detection
[6] Execute module RandDetect
RETURN
EXIT

FIGURE 2.5-14. RFSensorProbDetect Processing Logic.

<u>Block 1.</u> Attempt to detect radar signals from the target. If the radar is within the operating frequency of the passive sensor then check field of view and terrain masking.

Block 2. Determine the probability of detecting the target emitter. If the sensors probability of detection against the target times the target susceptibility to detection by the sensor type is greater than a normally distributed random number then the detection outcome is flagged as successful.

Block 3. Attempt to detect jammer signals from the target. If the jammer is active and within the operating band of the passive sensor it is a valid candidate for detection. Check field of view constraints and terrain masking.

Block 4. Determine the probability of detecting the target emitter. If the sensors probability of detection against the target times the target susceptibility to detection by the sensor type is greater than a normally distributed random number then the detection outcome is flagged as successful.

Block 5. Attempt to detect communications signals from the target. If the communications device is active and within the operating band of the passive sensor it is a valid candidate for detection. Check field of view constraints and terrain masking.

Block 6. Determine the probability of detecting the target emitter. If the sensors probability of detection against the target times the target susceptibility to detection by the sensor type is greater than a normally distributed random number then the detection outcome is flagged as successful.

Emitter - The module Emitter determines whether an target has any radar, jammer, or communications device emissions. If no emissions within the receiver band of the passive sensor are detected the target is not a candidate for detection. If the target has current emissions in the passive sensors receiver band then the target is eligible for detection by the passive sensor. Inputs and outputs for Emitter are listed in Table 2.5-6 and the steps performed by the module are illustrated in Figure 2.5-15.

TABLE 2.5-6. Inputs and Outputs for Emitter.

Inputs:	
Target	Pointer to target data
SenFreqLo	Sensor low frequency
SenFreqHi	Sensor high frequency
Outputs:	
Flag	Flag indicating whether valid emitter found

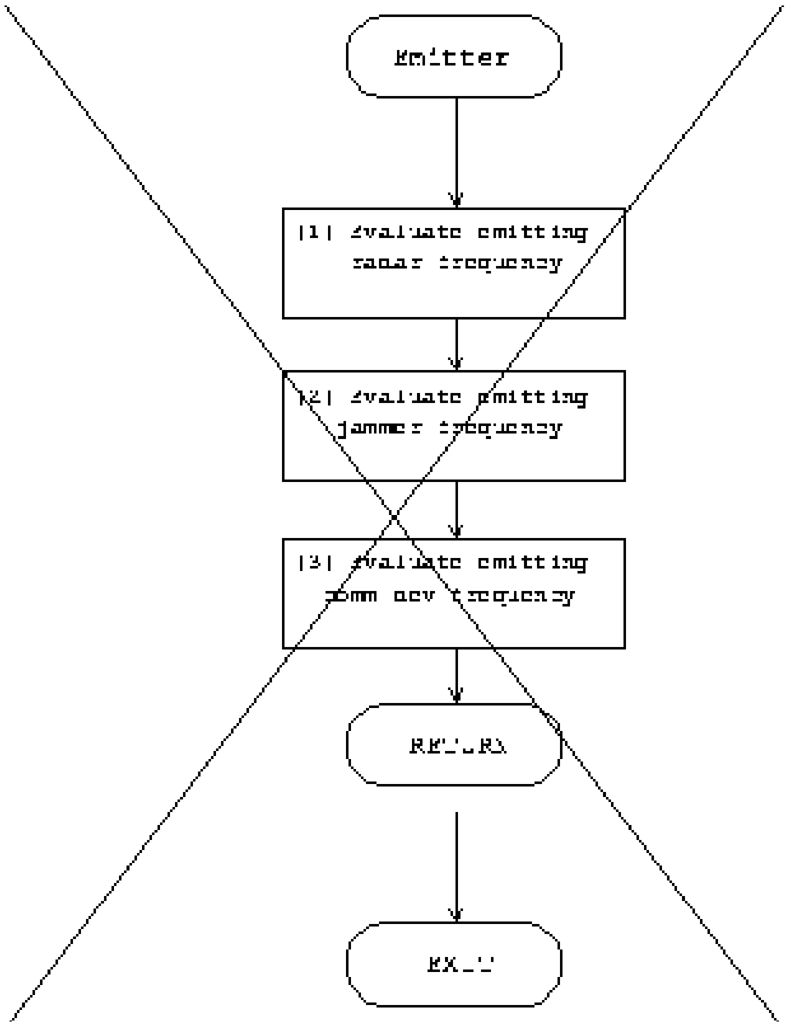
Emitter
[1] evaluate emitting radar frequency
[2] evaluate emitting jammer frequency
[3] evaluate emitting comm dev frequency
RETURN
EXIT

FIGURE 2.5-15. Emitter Processing Logic.

<u>Block 1.</u> If the target has radars. Loop over the emitting sensors until a match in frequency band is found. If a match is not found among all radars proceed to the next item.

<u>Block 2.</u> If the target has jammers. Loop over the emitting jammers until a match in frequency band is found. If a match is not found among all jammers process to the next emitter type.

<u>Block 3.</u> If the target has communication devices. Loop over the emitting communication devices until a match in frequency band is found. If a match is not found the target cannot be detected by the passive sensor.

ComputeBandRatio - The module ComputeBandRatio determines the bandwidth overlap between the passive sensor and the emitter it is attempting to detect. The bandwidth overlap ratio is used to adjust the signal power at the passive sensor antenna when an communication device or jammer is being evaluated. This signal power adjustment accounts for the communication device or jammer power being averaged over the entire operating frequency therefore reducing the power coming from any one channel to an average power level. Inputs and outputs for ComputeBandRatio are listed in Table 2.5-7 and the steps performed by the module are illustrated in Figure 2.5-16.

TABLE 2.5-7. Inputs and Outputs for ComputeBandRatio.

Inputs:	
RxFreqHi	Receiver high frequency
RxFreqLo	Receiver low frequency
RxBandwidth	Receiver operating bandwidth
RxNomFreq	Receiver nominal frequency
EmFreqHi	Emitter high frequency
EmFreqLo	Emitter low frequency
EmBandwidth	Emitter operating bandwidth
Outputs:	
EmNomFreq	Emitter nominal frequency
BRatio	Bandwidth overlap ratio

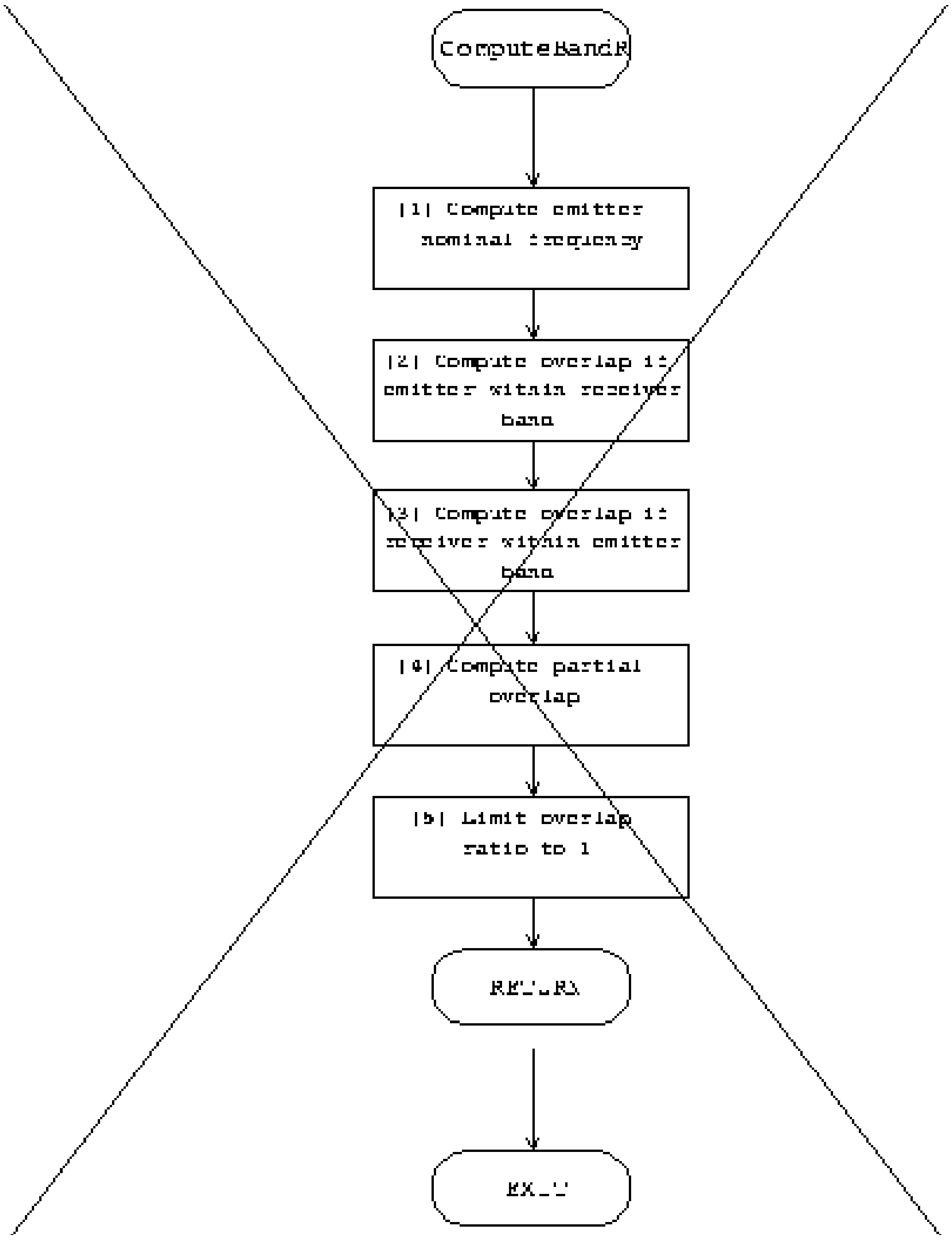
ComputeBandR
[1] Compute emitter nominal frequency
[2] Compute overlap if emitter within receiver band
[3] Compute overlap if receiver within emitter band
[4] Compute partial overlap
[5] Limit overlap ratio to 1
RETURN
EXIT

FIGURE 2.5-16. ComputeBandRatio Processing Logic.

Block 1. Compute the emitter nominal frequency and the bandwidth overlap ratio.

Block 2. If the receiver bandwidth is contained within the emitter operating frequency the bandwidth ratio equals the ratio of the receiver bw/emitter bw.

Block 3. If the emitter bandwidth is contained within the receiver bandwidth the bandwidth ratio equals 1.

Block 4. Compute the partial overlap.

Block 5. Limit the overlap ratio to 1.0.

GetSusceptibility - The module GetSusceptibility determines the susceptibility of the target to being detected based on the type of sensor which is attempting to detect it. The susceptibility inputs are user defined at the System level for most sensor types. Sensor types which are not explicitly defined use the susceptibility numbers of the general class to which it belongs. There are no explicit susceptibility inputs for the passive sensor type. However, there are susceptibility inputs for the Signal intelligence sensor type. The passive sensor was designed to mimic and improve on the SIGINT sensor model. As such the susceptibility inputs for the SIGINT sensor type apply to the passive sensor as well. The use of the system/sensor based susceptibility is limited to the probabilistic detection methodology. Inputs and outputs for GetSusceptibility are listed in Table 2.5-8 and the steps performed by the module are illustrated in Figure 2.5-17.

TABLE 2.5-8. Inputs and Outputs for GetSusceptibility.

Inputs:	
Scenario	Pointer to Scenario data
Target	Pointer to target platform
Outputs:	
Suscept	Sensor/System based detection susceptibility

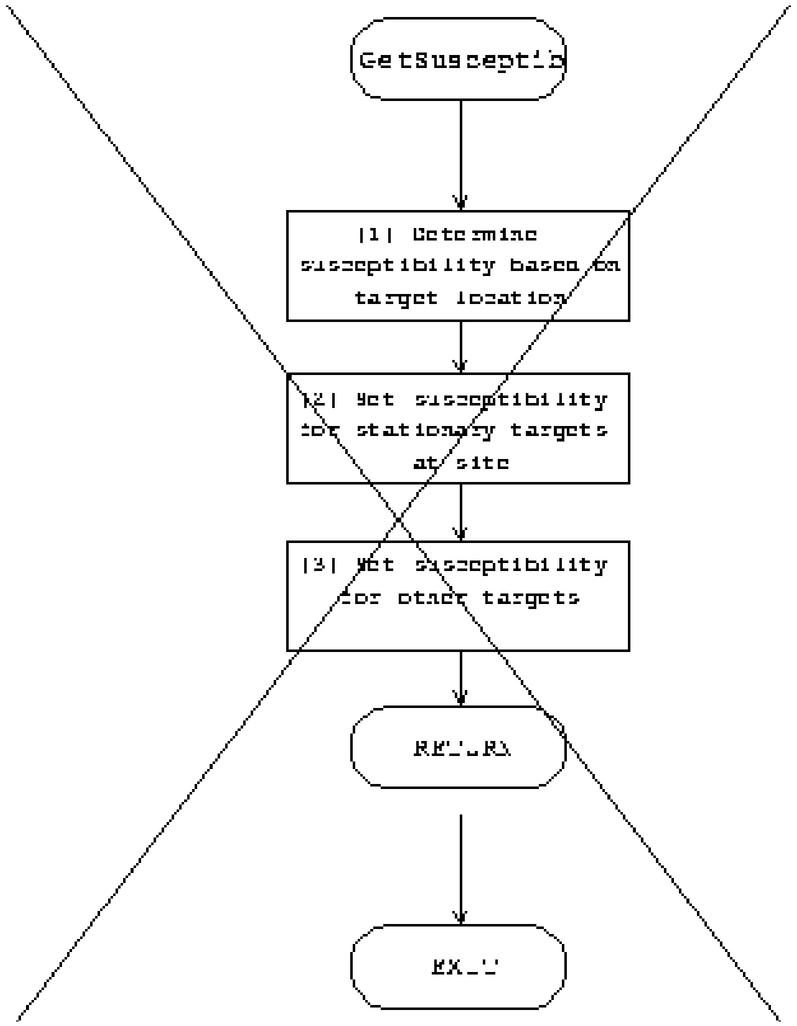
GetSusceptib
[1] Determine susceptibility based on target location
[2] Get susceptibility for stationary targets at site
[3] Get susceptibility for other targets
RETURN
EXIT

FIGURE 2.5-17. GetSusceptibility Processing Logic.

<u>Block 1</u>. Determine the target susceptibility to signal intelligence.

<u>Block 2</u>. If the target is at a hide site, reload site, etc... the susceptibility is defined by the susceptibility of the target while at the specific site. If the target is in transit the default susceptibility will apply.

<u>Block 3</u>. If the target is not at a site then the default susceptibility value applies.

2.5.4 Assumptions and Limitations

TABLE 2.5-9. Functional Element Limitations and Conditions of Applicability.

Functional Element	Limitations	Conditions of Applicability
2.1 Sensors	• All detections are assumed to be constrained by a common set of requirements: - Sensor platform and target are active - The target is within the sensor's FOV - The LOS between the target and sensor is not blocked by terrain	Sensors include IR, radar, HUMINT, IMINT, SIGINT, Launch Detection and Passive RF (Passive RF applies to version 5.00 only)
3.2.1.1 Radar	• Detections can be probabilistic or deterministic. Probabilistic detections are a function of SNR. Deterministic detections compare SNR to a threshold. • SNR is computed from the radar range equation. The receiver can be an ideal matched filter or the user can specify a receive processing loss • Peak SNR is used for all calculations • Radar resource management is a function of occupancy and duty cycle only; it does not model pulse scheduling. • Non-coherent pulse integration supports Swerling 0 through Swerling 4 target models • Coherent integration is assumed for Swerling 0, 1 and 3 targets • Coherent gain is not applied to Swerling 2 and 4 targets	Ground target detections are modeled probabilistically. Airborne targets can be modeled probabilistically or deterministically
3.2.1.4 Passive RF	Detection is probabilistic	Applies to ARM targeting decisions, jamming decisions and general signal intelligence collection.

TABLE 2.5-9. Functional Element Limitations and Conditions of Applicability.

Functional Element	**Limitations**	**Conditions of Applicability**
3.2.1.1 Radars	• Propagation factors for computing SNR are point values independent of radar/target location. (The addition of multipath/ diffraction and atmospheric model to version 5.00 nulls this statement)	• Applies to version 4.01
	• The detailed antenna model for phased array antennas does not represent the changes in sidelobe gains and shapes caused by beam pointing	
	• Antenna polarization is limited to vertical and horizontal	• Circular and elliptical polarizations cannot be represented